# Analytically Learning Variational Auto-Encoders

**Paul Christiano**[*] **David Matolcsi** david@alignment.org **George Robinson** george@alignment.org

## Abstract

Is it possible to understand the internal behaviour of a model without sampling inputs and passing those inputs through the model? We present a framework in which we can learn a representation for the distribution of activations in a simple neural network. This representation takes the form of a variational auto-encoder with specific assumptions on the family of distributions which we fit to the model via the encoder and decoder. Our contribution is to show that the loss of this specific autoencoder can be computed analytically, and so sampling the network is not required to minimize the representation loss.

## 1 Introduction

An auto-encoder is a simple neural network typically designed to learn efficient representations of data - which during interpretability tasks is often the activations of some layer of a larger neural network. The first part of the auto-encoder is an encoder and the second part is a decoder (each of which may be one or more layers deep), and the layer in between these is typically referred to as the latent space, or feature space, of the auto-encoder. These two parts are trained together to recover the identity function in such a way that the latent space carries a simplified representation of the data. Perhaps the best classical comparison to this is data compression (encoding) and reconstruction (decoding), such that compressed data carries more efficient information per bit than the original data.

Clearly, there must be some way of ensuring that the latent representation that we learn is 'efficient'. In the case of Sparse Auto-encoders (SAEs), the loss function is designed such that each input is described by just a few latent features - this has been used to identify useful feature directions in the intermediate layers of a language model that may correspond to human-interpretable concepts (for example see [2]). In the case of Variational Auto-encoders (VAEs, first defined in [5]), latent distributions are learned that can then be sampled to produce a generative model (for example in image generation, [4]). Both of these constructions train good latent representations by sampling the network and minimising the reconstruction loss of the generative part of the auto-encoder. However, sampling based methods are clearly not sufficient to capture behaviour of the model in events with sufficiently low probability in the training samples. It is possible that even some events with extremely low probability are of importance to model accurately. This leads us to the question: how can we learn latent representations without relying on sampling?

In a restricted setting, we propose a strategy for learning a good latent representation for the activations in the style of a VAE without sampling. We give an analytic calculation of the evidence lower bound (ELBO) loss that is standard practice for VAEs. In order for this to be analytically tractable, we make an assumption on the inital distribution of activations to the neural network, as well as a simple form for the transition functions: each layer of the network adds at most one direction of non-linearity (see equation (1) of Section 2.1). We then apply a VAE to each layer of this network which models the distribution on that layer as a sum of independent features with Gaussian noise. Our assumption on the initial distribution of activations is that it is given exactly by this form.

[*]Work done while at the Alignment Research Center prior to April 2024.

While the applicability of this construction is currently fairly narrow, we believe that more general architectures similar to this should allow us to better model rare events. This type of mechanistic analysis of the structure of a neural network has occurred in other areas (for example [3] for verifiable adversarial robustness), but our result appears to be the first use of the analytic approach to learn latent representations in a neural network.

## 2 Method

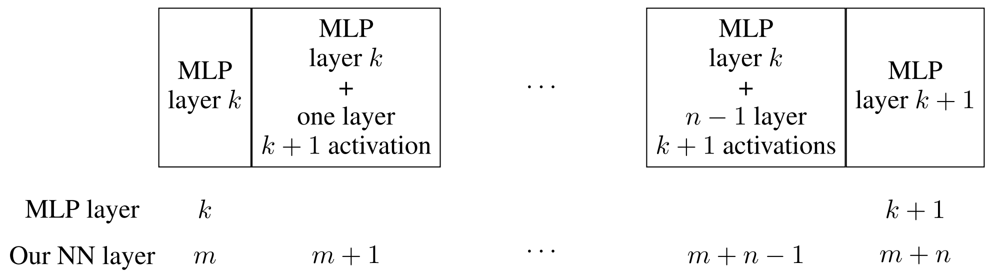
In this section, we will precisely define the structure of the neural network that we are studying, as well as the proposed architecture and distributions of the VAEs that we can analytically learn. We also derive the lower bound objective function in this model, expanding this particularly in the independent features case.

### 2.1 Problem scenario

Consider a neural network with $L$ layers with $n_1, ..., n_L$ neurons in each layer, and where the activations for each layer is related to the previous activations by a function of the form

$$\begin{aligned} &\mathbf{f}_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}} \\ &\mathbf{f}_i(\mathbf{x}_i) = \mathbf{A}_i\mathbf{x}_i + \mathbf{w}_i\psi_i(\langle \mathbf{A}_i\mathbf{x}_i, \mathbf{u}_i\rangle) \end{aligned} \tag{1}$$

where $\mathbf{A}_i \in \mathbb{R}^{n_{i+1}\times n_i}, \mathbf{x}_i \in \mathbb{R}^{n_i}, \mathbf{u}_i, \mathbf{w}_i \in \mathbb{R}^{n_{i+1}}$ and $\psi_i$ is a quick-to-compute non-linear function, such as ReLU or sigmoid (which may vary layer to layer). While this is a simple functional form, any Multi-Layer Perceptron (MLP) or transformer network can be built from these simple steps (although to perform the attention circuit for a single layer of a transformer would take many layers with this method). For example, to reconstruct a single transition between two width $n$ layers of an MLP, we could reproduce each activation one by one in an auxiliary direction while remembering the previous activations, and in the final step project to forget the previous layer and compute the final activation.



The problem we are concerned with in this paper is, given the input distribution of activations in $\mathbb{R}^{n_1}$, how can we approximate the distribution of activations in $\mathbb{R}^{n_i}$. As discussed above, we will learn a VAE representation of the activations at each layer, and this will be learned inductively from the VAE modelling the previous layer. Therefore, we will now restrict ourselves to just the case of $\mathbf{f} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$.

#### 2.1.1 The Network Architecture

More precisely, our set-up is as follows: consider two random variables $\mathbf{x}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{n_2}$. Assume that samples from $\mathbf{x}_1$ are generated by a VAE in the following process from a latent random variable $\mathbf{z}_1$:

1. Sample from a distribution $P_1(\mathbf{z}_1)$ called the *prior*; and then,
2. Generate $\mathbf{x}_1$ from a conditional distribution $P_1(\mathbf{x}_1|\mathbf{z}_1)$ called the *likelihood*.

We assume that the prior and likelihood distributions are tractable and analytic, however the marginal $P_1(\mathbf{x}_1)$ and posterior $P_1(\mathbf{z}_1|\mathbf{x}_1)$ are then implicitly defined (via an integral over $\mathbf{z}_1$) and so are infeasible to compute quickly. In addition suppose that $\mathbf{x}_2$ is constructed as

$$\mathbf{x}_2 = \mathbf{f}(\mathbf{x}_1) = \mathbf{A}\mathbf{x}_1 + \psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle)\mathbf{w}.$$

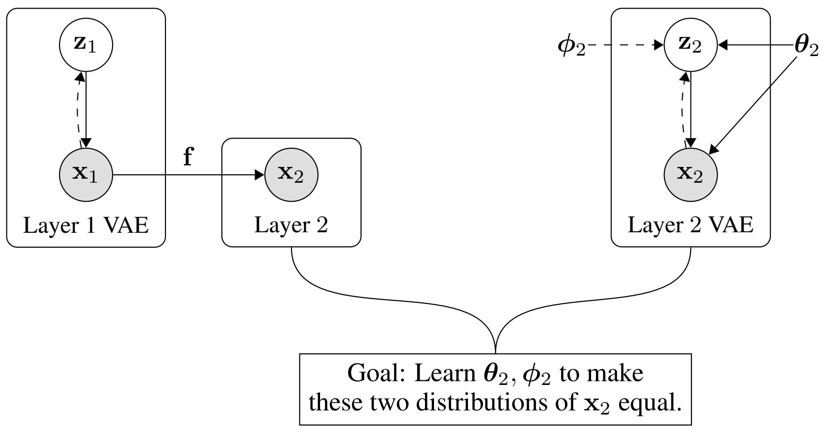


Figure 1: On the left, we have the distribution on layer 1 modelled by a VAE trained in the previous step (or for the initial step this is the distributional assumption we make on the inputs). Applying $\mathbf{f}$ gives a distribution on layer 2 that we wish to model. We do this using the VAE model on the right. For parameter $\boldsymbol{\theta}_2, \boldsymbol{\phi}_2$ we will be able to analytically compute the variational upper bound for the KL divergence between these two distributions on the second layer and therefore be able to train the VAE on layer 2 without sampling. Dashed lines denote the approximate posterior distributions (the encoders).

We are interested in the following problem:

> Suppose that we wish to find parameters $\boldsymbol{\theta}$ so that the VAE process above with distributions $P_{2,\boldsymbol{\theta}}(\mathbf{z}_2)$ and $P_{2,\boldsymbol{\theta}}(\mathbf{x}_2|\mathbf{z}_2)$ has marginal $P_{2,\boldsymbol{\theta}}(\mathbf{x}_2)$ that closely approximates $\mathbf{f}(P_1(\mathbf{x}_1))$.

Notice that, as for $P_1$, we do not directly have access to $P_{2,\boldsymbol{\theta}}(\mathbf{x}_2)$ (or indeed to $\mathbf{f}(P_1(\mathbf{x}_1))$) and so we require an *indirect route* to minimize the approximation loss between these two distributions. As in [5], the ability to train such latent representations without direct access to the marginals comes from a probabilistic encoder $Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x}_2)$ which provides an analytically tractable approximation to the intractable posterior $P_{2,\boldsymbol{\theta}}(\mathbf{z}_2|\mathbf{x}_2)$ (note the potentially misleading notation here - there in fact is no joint distribution $Q_{2,\boldsymbol{\phi}}(\mathbf{x}_2, \mathbf{z}_2)$, we are simply assigning a distribution over latents to each value of $\mathbf{x}_2$).

In order to optimally approximate $\mathbf{f}(P_1(\mathbf{x}_1))$ with $P_2(\mathbf{x}_2)$, our aim is to minimize the cross-entropy, which equals

$$H((\mathbf{f}(P_1))(\mathbf{x}_2), P_{2,\boldsymbol{\theta}}(\mathbf{x}_2)) = -\mathbb{E}_{\mathbf{x}_1 \sim P_1}\left[\log\left(P_{2,\boldsymbol{\theta}}\left(\mathbf{f}(\mathbf{x}_1)\right)\right]\right. \tag{2}$$

where $\mathbf{f}(P_1)$ denotes the pushforward of the $P_1(\mathbf{x}_1)$ distribution by the function $\mathbf{f}$.

By fitting $P_{2,\boldsymbol{\theta}}(\mathbf{z}_2), P_{2,\boldsymbol{\theta}}(\mathbf{x}_2|\mathbf{z}_2), Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x}_2)$ in parallel, we can minimize a variational upper bound for this cross entropy. The result of the optimisation over $\boldsymbol{\theta}, \boldsymbol{\phi}$ is two-fold:

1. Approximation of the parameter $\boldsymbol{\theta}$ will allow us to generate data from a distribution closely resembling $\mathbf{f}(P_1(\mathbf{x}_1))$. Note that the parameters in $\boldsymbol{\theta}$ which are used for $P_{2,\boldsymbol{\theta}}(\mathbf{z}_2)$ and $P_{2,\boldsymbol{\theta}}(\mathbf{x}_2|\mathbf{z}_2)$ may be distinct, but we group them together into $\boldsymbol{\theta}$ since they are both parameters for the decoder.
2. Approximation of the parameter $\boldsymbol{\phi}$ will allow us to infer the latent variable $\mathbf{z}_2$ from an observed $\mathbf{x}_2$.

## 2.2 The Evidence Lower Bound

The simple but fundamental result of variational inference is the Evidence Lower Bound (ELBO). In words, this tells us that computing $P(\mathbf{x})$ using the approximate posterior is only going to underestimate the probability, and it does so by a factor depending on how different the approximate posterior is to the true posterior. This difference of probability distributions is measured by Kullback-Leibler (KL) divergence, which for two distributions $P, Q$ is given by

$$D_{KL}(P||Q) := \mathbb{E}_{\mathbf{x}\sim P}\left[\log\left(\frac{P(\mathbf{x})}{Q(\mathbf{x})}\right)\right]$$

and has the property that $D_{KL}(P||Q) \geq 0$ with equality if and only if $P = Q$.

**Lemma 2.1.** For a fixed $\mathbf{x}$, given a joint distribution $P(\mathbf{x}, \mathbf{z})$ and a distribution $Q(\mathbf{z})$,

$$\log P(\mathbf{x}) \geq -D_{KL}\left(Q(\mathbf{z})||P(\mathbf{z})\right) + \mathbb{E}_{Q(\mathbf{z})}\left[\log P(\mathbf{x}|\mathbf{z})\right] \tag{3}$$

with equality if and only if the distribution $Q(\mathbf{z})$ is precisely equal to the posterior distribution $P(\mathbf{z}|\mathbf{x})$.

*Proof.* According to Bayes-theorem, for every $\mathbf{z}$,

$$P(\mathbf{x}) = \frac{P(\mathbf{z})P(\mathbf{x}|\mathbf{z})}{P(\mathbf{z}|\mathbf{x})}$$

Taking the logarithm of this expression and introducing the distribution $Q(\mathbf{z})$ as an approximation to $P(\mathbf{z}|\mathbf{x})$, we get

$$\begin{aligned}\log P(\mathbf{x}) &= \log\left(\frac{Q(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})}\frac{P(\mathbf{z})}{Q(\mathbf{z})}P(\mathbf{x}|\mathbf{z})\right)\\ &= \log\left(\frac{Q(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})}\right) - \log\left(\frac{Q(\mathbf{z})}{P(\mathbf{z})}\right) + \log P(\mathbf{x}|\mathbf{z}).\end{aligned}$$

Now the expectation over $\mathbf{z} \sim Q$ of this expression becomes

$$\log P(\mathbf{x}) = D_{KL}\left(Q(\mathbf{z})||P(\mathbf{z}|\mathbf{x})\right) - D_{KL}\left(Q(\mathbf{z})||P(\mathbf{z})\right) + \mathbb{E}_{Q(\mathbf{z})}\left[\log P(\mathbf{x}|\mathbf{z})\right],$$

using the definition of KL divergence. The inequality follows from the fact that KL divergence $D_{KL}(Q(\mathbf{z})||P(\mathbf{z}|\mathbf{x}))$ is non-negative and zero if and only the two distributions are equal. □

Notice in particular that the discrepancy in the bound is precisely the KL divergence of the decoder $Q(\mathbf{z})$ with the true posterior $P(\mathbf{z}|\mathbf{x})$, and so maximizing the RHS of equation (3) will maximize $\log P(\mathbf{x})$ as well as fitting the decoder to the true posterior.

Applying Lemma 2.1 to our model (specifically Equation (2)) and taking the expectation over $\mathbf{x}_1 \sim P_1$, we arrive at the following:

**Lemma 2.2.**

$$\begin{aligned}-\mathbb{E}_{P_1(\mathbf{x}_1)}\left[\log P_{2,\boldsymbol{\theta}}(\mathbf{f}(\mathbf{x}_1))\right] \leq \mathbb{E}_{P_1(\mathbf{x}_1)}&\left[D_{KL}\left(Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))||P_{2,\boldsymbol{\theta}}(\mathbf{z}_2)\right)\right]\\ &- \mathbb{E}_{\substack{P_1(\mathbf{x}_1)\\ Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}}\left[\log P_{2,\boldsymbol{\theta}}(\mathbf{f}(\mathbf{x}_1)|\mathbf{z}_2)\right].\end{aligned} \tag{4}$$

*Proof.* From Lemma 2.1, we get for any $\mathbf{x}_1$,

$$-\log P_{2,\boldsymbol{\theta}}\left(\mathbf{f}(\mathbf{x}_1)\right) \leq D_{KL}\left(Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))||P_{2,\boldsymbol{\theta}}(\mathbf{z}_2)\right) - \mathbb{E}_{Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}\left[\log P_{2,\boldsymbol{\theta}}(\mathbf{f}(\mathbf{x}_1)|\mathbf{z}_2)\right].$$

Now we take expectations with respect to $\mathbf{x}_1 \sim P_1(\mathbf{x}_1)$. □

**Definition 2.3.** Let us write

$$\mathcal{L}_{\mathbf{x}_2} = \mathcal{L}_{\mathbf{x}_2}(\boldsymbol{\theta}, \boldsymbol{\phi}) := D_{KL}\left(Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x}_2)||P_{2,\boldsymbol{\theta}}(\mathbf{z}_2)\right) - \mathbb{E}_{Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x}_2)}\left[\log P_{2,\boldsymbol{\theta}}(\mathbf{x}_2|\mathbf{z}_2)\right]$$

for the *VAE loss contribution* from $\mathbf{x}_2$, and

$$\mathcal{L} = \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) := \mathbb{E}_{P_1(\mathbf{x}_1)}\left[\mathcal{L}_{\mathbf{f}(\mathbf{x}_1)}\right]$$

for the *VAE loss*. This equals the RHS of Equation (4).

This simple lemma gives an upper bound, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$, for the approximation loss of the pushforward distribution $\mathbf{f}(P_1(\mathbf{x}_1))$ using the latent model in terms of *computable* quantities (in particular using the decoder). The method will therefore proceed by minimizing the RHS of equation (4) with respect to the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$.

## 2.3 Relation to Sparse Autoencoders

An SAE is similar to a VAE in construction but instead of the encoder and decoder being distributions, they are simply functions in the following way:

1. each vector of activations $\mathbf{x} \in \mathbb{R}^n$ is encoded as a combination of learned features $\mathbf{e}(\mathbf{x}) \in \mathbb{R}^M_{\geq 0}$, where typically $M \gg n$;
2. each feature vector $\mathbf{e} \in \mathbb{R}^M$ is decoded to a reconstruction for the activations, $\widehat{\mathbf{x}}(\mathbf{e}) \in \mathbb{R}^n$;
3. The functions $\mathbf{e}$ and $\widehat{\mathbf{x}}$ belong to a parameterized family and training the SAE consists of learning these parameters.

The learning takes place using a loss function generally of the form

$$\mathcal{L}_{SAE,S}(\mathbf{x}) := ||\mathbf{x} - \widehat{\mathbf{x}}(\mathbf{e}(\mathbf{x}))||_2^2 + \lambda S(\mathbf{e}(\mathbf{x}))$$

where $S$ is a function of the feature coefficients designed to encourage sparse feature decomposition, and $\lambda$ is a sparsity coefficient that sets the balance between reconstruction loss and sparsity. The choice of sparsity loss gives some flexibility in the framework of an SAE - most commonly an $L_1$ sparsity is chosen, but other losses are occasionally used.

**Lemma 2.4.** The SAE loss function, $\mathcal{L}_{SAE,||\cdot||_1}(\mathbf{x})$, is equal (up to an affine transformation) to the VAE loss contribution, $\mathcal{L}_{\mathbf{x}}$, when we assume that

- The VAE encoder is given as a normal distribution in terms of the SAE encoder by a point distribution $Q_{2,\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \delta_{\mathbf{e}(\mathbf{x})}(\mathbf{z})$;
- The VAE decoder is given as an isotropic normal distribution $P_{2,\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\widehat{\mathbf{x}}(\mathbf{z}), \rho)$ around the SAE decoder;
- The VAE prior is given by a product of exponential distributions $P_{2,\boldsymbol{\theta}}(\mathbf{z}) = \prod_{j=1}^M \mu \exp(-\mu z_j)$;
- The standard deviation $\rho$ of the decoder and the mean $\mu$ of the prior are related to the sparsity parameter by
$$2\mu\rho = \lambda$$

*Proof.* Under these assumptions, the first term of $\mathcal{L}_{\mathbf{x}}$ is

$$D_{KL}\left(\delta_{\mathbf{e}(\mathbf{x})}(\mathbf{z})||\prod_{j=1}^M \mu \exp(-\mu z_j)\right) = -\log\left(\prod_{j=1}^M \mu \exp(-\mu e_j(\mathbf{x}))\right) = -M\log(\mu) + \mu\sum_{j=1}^M e_j(\mathbf{x}),$$

where we use the fact that $D_{KL}(\delta_y(\mathbf{x})||P(\mathbf{x}))$ is typically interpreted as $-\log P(y)$ (equivalently the entropy of a delta distribution is considered to be zero, as can be seen by taking the limiting entropy of a family of distributions approaching the delta distribution). The second term of $\mathcal{L}_{\mathbf{x}}$ is

$$-\mathbb{E}_{\mathbf{z}\sim\delta_{\mathbf{e}(\mathbf{x})}}\left[\log\left(\frac{1}{(2\pi\rho)^{M/2}}\exp(-\frac{1}{2\rho}||\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{z})||_2^2)\right)\right] = \frac{M}{2}\log(2\pi\rho) + \frac{1}{2\rho}||\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{e}(\mathbf{x}))||_2^2,$$

where the quantity inside the logarithm is the pdf of the normal distribution. Summing these two terms we get

$$\begin{aligned}\mathcal{L}_{\mathbf{x}} &= \frac{M}{2}\log\left(\frac{2\pi\rho}{\mu^2}\right) + \frac{1}{2\rho}||\mathbf{x}-\widehat{\mathbf{x}}(\mathbf{e}(\mathbf{x}))||_2^2 + \mu\sum_{j=1}^M e_j(\mathbf{x}) \\ &= \frac{M}{2}\log\left(\frac{2\pi\rho}{\mu^2}\right) + \frac{1}{2\rho}\mathcal{L}_{SAE,||\cdot||_1}(\mathbf{x}).\end{aligned}$$

Note that since $\mathbf{e}(\mathbf{x})_j \geq 0$, the summation in the first line is indeed equal to the $L_1$-norm of $\mathbf{e}(\mathbf{x})$. Notice that the normal VAE decoder is responsible for the mean squared error (reconstruction loss) term of the SAE loss, and the exponential VAE prior is responsible for the $||\cdot||_1$-sparsity penalty term of the SAE loss. $\square$

We can see from this proof that a similar lemma holds for any sparsity penalty, $S$, provided that $\exp(-S(\mathbf{z}))$ has finite integral over $\mathbf{z} \in \mathbb{R}^M_{\geq 0}$. In addition, the corresponding VAE prior will factor as as product distribution over the coordinates of $\mathbf{z}$ if and only if the sparsity penalty is of the form $S(\mathbf{z}) = \sum_{j=1}^M S_j(z_j)$.

### 2.4 Assuming independent latents

In this section, we give an explicit example of distributional families for the $P_1(\mathbf{z}_1), P_2(\mathbf{z}_2)$ latent distributions and the $P_1(\mathbf{x}_1|\mathbf{z}_1), P_2(\mathbf{x}_2|\mathbf{z}_2)$ decoders and the $Q_2(\mathbf{z}_2|\mathbf{x}_2)$ encoders, such that the $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ loss from Lemma 2.2 can be analytically calculated (which will be done in the following section).

We will assume that any function we use is Lipschitz and any one-dimensional marginal of any distribution we deal with is supported on a bounded interval of length at most $T$. Thus, we can approximate any one-dimensional distribution by representing it as fixing $\frac{T}{\varepsilon}$ equally spaced bins of $\varepsilon$-length, and determining the probability mass in each bin and treating the distribution as discrete on $\frac{T}{\varepsilon}$ equally spaced points.

The error we accumulate with this approximation will be $O(\varepsilon)$ thanks to the Lipschitz-property, so if we choose $\varepsilon$ to be small enough, this approximation can be very fine-grained.

Other than this, we make the following assumptions about the setting:

1. Both $P_1(\mathbf{z}_1)$ and $P_2(\mathbf{z}_2)$ factor as product distributions over the coordinates of $\mathbb{R}^{m_1}$ and $\mathbb{R}^{m_2}$ respectively, so that
$$P_1(\mathbf{z}_1) = \prod_{j=1}^{m_1} P_{1,j}(\alpha_j), \quad P_2(\mathbf{z}_2) = \prod_{j=1}^{m_2} P_{2,j}(\beta_j).$$
for $\mathbf{z}_1 = (\alpha_1, \alpha_2, \ldots, \alpha_{m_1})$ and $\mathbf{z}_2 = (\beta_1, \beta_2, \ldots, \beta_{m_2})$. (Note the change of letter for the coordinates to avoid the over use of subscripts.)
2. For every value of $\mathbf{x}_2$, the distribution $Q_2(\mathbf{z}_2|\mathbf{x}_2)$ factors as a product over coordinates
$$Q_2(\mathbf{z}_2|\mathbf{x}_2) = \prod_{j=1}^{m_2} Q_{2,j}(\beta_j|\mathbf{x}_2).$$
3. There exist linear maps $\mathbf{B}_1 : \mathbb{R}^{m_1} \to \mathbb{R}^{n_1}$ and $\mathbf{B}_1 : \mathbb{R}^{m_2} \to \mathbb{R}^{n_2}$ such that
$$P_1(\mathbf{x}_1|\mathbf{z}_1) = \mathcal{N}(\mathbf{B}_1\mathbf{z}_1, \sigma_1^2\mathbf{I}), \quad P_2(\mathbf{x}_2|\mathbf{z}_2) = \mathcal{N}(\mathbf{B}_2\mathbf{z}_2, \sigma_2^2\mathbf{I})$$
are isotropic normal distributions.
4. For all $j$, the $Q_{2,j}(\beta_j|\mathbf{x}_2)$ distribution is defined by just $t$ parameters that linearly depend on $\mathbf{x}_2$:
$$Q_{2,j}(\beta_j|\mathbf{x}_2) = Q'_{2,j}(\beta_j|\mathbf{C}_j\mathbf{x}_2)$$
such that
   - $\mathbf{C}_j \in \mathbb{R}^{n_2 \times t}$ matrix maps $\mathbf{x}_2$ into an $\mathbb{R}^t$ parameter-space where $t$ is small.
   - It takes just $O(\frac{T}{\varepsilon})$ time to calculate the full distribution of $Q'_{2,j}(\beta_j|\mathbf{x}_2)$ up to the usual $\frac{T}{\varepsilon}$ bins if we are given the $\mathbf{C}_j\mathbf{x}_2$ parameters.

   (Example: $Q_{2,j}(\beta_j|\mathbf{x}_2) = Q'_{2,j}(\beta_j|\mathbf{C}_j\mathbf{x}_2)$ is always an exponential distribution with $\mathbf{C}_j\mathbf{x}_2 \in \mathbb{R}^1$ being the parameter of the exponential.)

A few important things to notice:

a) The way the $P_1$ input distribution is given to us (the sum of independent features plus Gaussian noise), is the same format in which we model the $P_2$ distribution. This means that after we get a model of the $P_2$ distribution, we can use that model as an input to model the third layer in the same way, and so on, modelling every layer of a neural network going forward.

b) We have suppressed the suffixes $\boldsymbol{\theta}, \boldsymbol{\phi}$, but in reality $(\{P_{2,j}\}, \mathbf{B}_2, \sigma_2)$ are the parameters of $\boldsymbol{\theta}$ and $(\{\mathbf{C}_j\})$ are the parameters of $\boldsymbol{\phi}$.

## 3 Analytic Computation

As the main result of our paper, we will prove the following theorem to show that it is possible to calculate and minimize the VAE loss in this particular setting without any sampling required.

**Theorem 3.1.** Given the assumptions above, we can analytically calculate the value of the loss from Lemma 2.2,

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{P_1(\mathbf{x}_1)} \left[D_{KL}\left(Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))||P_2(\mathbf{z}_2)\right)\right] - \mathbb{E}_{\substack{P_1(\mathbf{x}_1) \\ Q_{2,\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}} [\log P_2(\mathbf{f}(\mathbf{x}_1)|\mathbf{z}_2)]$$

in time that is polynomial in $\frac{T}{\varepsilon}, m_1, m_2, n_1, n_2$.

**Remark 3.2.** It's worth noting that while the time complexity is polynomial in $\frac{T}{\varepsilon}, m_1, m_2, n_1, n_2$, the variable $t$ appears in the exponent, so the computation is only fast if $t$ is small, which corresponds to the $Q_{2,j}(\beta_j|\mathbf{x}_2)$ distributions being defined by only a few parameters.

Before we get into the proof, we will state a simple fact, then a Lemma that is foundational to all the rest of the proof.

**Fact 3.3** (Fast Convolutions)**.** Given vectors $\mathbf{d}_1, \mathbf{d}_2, \ldots \mathbf{d}_m \in \mathbb{R}^t$ and one-dimensional probability distributions $P_1, P_2, \ldots, P_m$, we define random variable $\mathbf{x}$ as

$$\mathbf{x} = \sum_{j=1}^{m} \alpha_j \mathbf{d}_j$$

where each $\alpha_j$ is sampled from $P_j$ independently, it takes $\widetilde{O}\left(m(\frac{T}{\varepsilon})^t\right)$ time to calculate the distribution of $\mathbf{x}$ up to the usual approximations, causing just an $O(m\varepsilon)$ error.

*Proof.* Since the $\alpha_j$ values are sampled independently, we just need to calculate the convolution of the $\alpha_j \mathbf{d}_j$ distributions. As in our usual approximations, we represent the distributions on $\mathbb{R}^t$ with each coordinate's value specified in $\frac{T}{\varepsilon}$ equally spaced bins. Thus, the whole distribution is defined on $(\frac{T}{\varepsilon})^t$ points.

We first calculate the convolution of the distributions of $\alpha_1 \mathbf{d}_1$ and $\alpha_2 \mathbf{d}_2$, then the convolution of this with the distribution of $\alpha_3 \mathbf{d}_3$ and so on. Each convolution takes only $O(N \log N)$ time in the size of the space using Fast Fourier Transform, and we do $m$ convolutions on the $(\frac{T}{\varepsilon})^t$ sized space, so altogether the operation takes $\widetilde{O}\left(m(\frac{T}{\varepsilon})^t\right)$ time.

□

Based on this fact, we prove the following central lemma that all the rest of our calculation relies on:

**Lemma 3.4.** For any collection of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t \in \mathbb{R}^{n_2}$, we can analytically calculate the joint distribution of $\{\langle \mathbf{f}(\mathbf{x}_1), \mathbf{v}_j\rangle\}_{1 \le j \le t}$ if $x_1$ is sampled from $P_1(\mathbf{x}_1)$. The time required is $\widetilde{O}\left(m_1(\frac{T}{\varepsilon})^{t+1}\right)$. In particular, it's possible to fully calculate the $t$-dimensional $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$ distributions in polynomial time.

*Proof.* We assumed that $\mathbf{f}$ takes the form of

$$\mathbf{f}(\mathbf{x}_1) = \mathbf{A}\mathbf{x}_1 + \mathbf{w}\psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle)$$

We only need to prove that we can calculate the joint distribution $D$ of $\{\langle \mathbf{A}\mathbf{x}_1, \mathbf{v}_j\rangle\}_{1 \le j \le t}$ and $\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle$ in $\widetilde{O}\left(m_1(\frac{T}{\varepsilon})^{t+1}\right)$ time.

This is because the desired $\{\langle \mathbf{f}(\mathbf{x}_1), \mathbf{v}_j \rangle\}_{1 \leq j \leq t}$ distribution is just the push-forward of $D$ by the the linear transformation

$$F : (a_1, a_2, \dots a_t, b) \to (a_1 + \langle \mathbf{w}, \mathbf{v}_1 \rangle \psi(b), a_2 + \langle \mathbf{w}, \mathbf{v}_2 \rangle \psi(b), \dots a_t + \langle \mathbf{w}, \mathbf{v}_t \rangle \psi(b))$$

function. As we track the distribution of $\langle \mathbf{A}\mathbf{x}_1, \mathbf{v}_j \rangle$ and $\langle \mathbf{A}\mathbf{x}_1, \mathbf{w} \rangle$ in discrete distributions on $\frac{T}{\varepsilon}$ bins, there are $(\frac{T}{\varepsilon})^{t+1}$ values on which $D$ is defined, so calculating the pushforward only takes $O((\frac{T}{\varepsilon})^{t+1})$ time.

As for calculating the distribution of $\{\langle \mathbf{A}\mathbf{x}_1, \mathbf{v}_j \rangle\}_{1 \leq j \leq t}$ and $\langle \mathbf{A}\mathbf{x}_1, \mathbf{w} \rangle$, let's call $\mathbf{W}$ the matrix whose rows are $\{\mathbf{v}_j\}$ and $\mathbf{w}$, and let's call $\mathbf{G} = \mathbf{W}\mathbf{A}$. We just want to calculate the distribution of $\mathbf{G}\mathbf{x}_1$.

We know that $P_1(\mathbf{x}_1|\mathbf{z}_1)$ is a normal distribution $\mathcal{N}(\mathbf{B}_1\mathbf{z}_1, \sigma_1^2\mathbf{I})$, i.e.

$$\mathbf{x}_1 = \mathbf{B}_1\mathbf{z}_1 + \boldsymbol{\zeta},$$

where $\boldsymbol{\zeta}$ is an isotropic Gaussian noise with mean 0 and standard deviation $\sigma_1$ that is independent of $\mathbf{z}_1$. This means that

$$\mathbf{G}\mathbf{x}_1 = \mathbf{G}\mathbf{B}_1\mathbf{z}_1 + \mathbf{G}\boldsymbol{\zeta}.$$

The $\alpha_j$ coordinates of $\mathbf{z}_1$ are independently sampled from the $P_{1,j}(\alpha_j)$ distributions. Let $\mathbf{d}_j \in \mathbb{R}^{t+1}$ denote the $j$th column of the matrix $\mathbf{G}\mathbf{B}_1$. This means that

$$\mathbf{G}\mathbf{B}_1\mathbf{z}_1 = \sum_{j=1}^{m_1} \alpha_j \mathbf{d}_j$$

where each $\alpha_j$ is sampled from $P_{1,j}$ independently.

Then, according to Fact 3.3 (Fast Convolutions), it takes $\widetilde{O}\left(m_1(\frac{T}{\varepsilon})^{t+1}\right)$ time to calculate the distribution of $\mathbf{G}\mathbf{B}_1\mathbf{z}_1$. Now we need to convolve this with the independent distribution of $\mathbf{G}\boldsymbol{\zeta}$ which is easy to calculate, as it is just a normal distribution with a given covariance matrix. The convolution again takes only $\widetilde{O}\left((\frac{T}{\varepsilon})^{t+1}\right)$ time.

This means that calculating the distribution of $\{\langle \mathbf{f}(\mathbf{x}_1), \mathbf{v}_j \rangle\}_{1 \leq i \leq t}$ indeed takes $\widetilde{O}\left(m_1(\frac{T}{\varepsilon})^{t+1}\right)$ time.

□

Now we are ready to present the proof of Theorem 3.1.

*Proof.* Using the fact that both the $P_2(\mathbf{z}_2)$ and $Q_2(\mathbf{z}_2||\mathbf{f}(\mathbf{x}_1))$ distributions are coordinate-wise independent, we can see that

$$\begin{aligned} D_{KL}\left(Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))||P_2(\mathbf{z}_2)\right) &= \mathbb{E}_{Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}[\log(Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))) - \log(P_2(\mathbf{z}_2))] \\ &= \sum_{j=1}^{m_2} \mathbb{E}_{Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))}\left[\log(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))) - \log(P_{2,j}(\beta_j))\right] \\ &= \sum_{j=1}^{m_2} D_{KL}\left(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))||P_{2,j}(\beta_j)\right) \end{aligned}$$

We can also observe that

$$P_2(\mathbf{x}_2|\mathbf{z}_2) = \left(\frac{1}{\sqrt{2\pi\sigma_2}}\right)^{n_2} \exp\left(-\frac{1}{2\sigma_2}||\mathbf{x}_2 - \mathbf{B}_2\mathbf{z}_2||_2^2\right), \tag{5}$$

since we have assumed that $P_2(\mathbf{x}_2|\mathbf{z}_2)$ is an isotropic Gaussian distribution.

Taken together, this means that our loss function takes the form

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) &= \mathbb{E}_{P_1(\mathbf{x}_1)}\left[D_{KL}\left(Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))||P_2(\mathbf{z}_2)\right)\right] - \mathbb{E}_{\substack{P_1(\mathbf{x}_1)\\ Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}}\left[\log P_2(\mathbf{f}(\mathbf{x}_1)|\mathbf{z}_2)\right]\\ &= \left(\sum_{j=1}^{m_2}\mathbb{E}_{P_1(\mathbf{x}_1)}\left[D_{KL}(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))||P_{2,j}(\beta_j))\right]\right) + \frac{n_2}{2}\log(2\pi\sigma_2)\\ &\quad + \frac{1}{2\sigma_2}\mathbb{E}_{\substack{P_1(\mathbf{x}_1)\\ Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}}\left[||\mathbf{f}(\mathbf{x}_1)-\mathbf{B}_2(\mathbf{z}_2)||_2^2\right] \qquad (6)\end{aligned}$$

We will calculate the terms of (6) one by one.

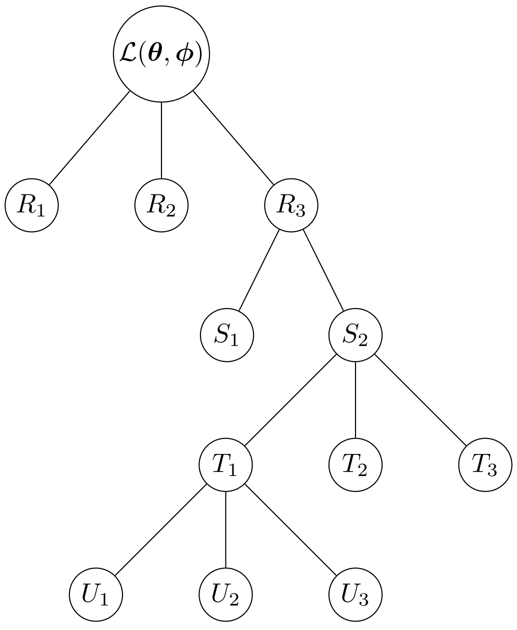


Figure 2: This is the diagram of the calculation we are executing. We break up $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi})$ to a sum of three terms, then we break up some of the terms further. We give names to each term, and each term is the sum of its children in the diagram. (Except for $R_3$, witch equals $\frac{1}{2\sigma_2}$ times the sum of its children.)

**First term of (6):** $R_1 = \sum_{j=1}^{m_2}\mathbb{E}_{P_1(\mathbf{x}_1)}\left[D_{KL}(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))||P_{2,j}(\beta_j)\right]$

For each $1 \le j \le m_2$, the $Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1)) = Q'_{2,j}(\beta_j|\mathbf{C}_j\mathbf{f}(\mathbf{x}_1))$ distribution's density function is calculable in $O(\frac{T}{\varepsilon})$ time from the value of $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$.

$$D_{KL}(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))||P_{2,j}(\beta_j)) = \sum_{\beta_j} Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))\log\left(\frac{Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))}{P_{2,j}(\beta_j)}\right)$$

is calculable in $O(\frac{T}{\varepsilon})$ time if the $P_{2,j}(\beta_j)$ and $Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))$ distributions are given, so it's calculable for every value of $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$.

According to Lemma 3.4, we can calculate the distributions of $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$ in polynomial time.

There are $(\frac{T}{\varepsilon})^t$ possible values of $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$, and we know the probability distribution of these values, so

$$\mathbb{E}_{P_1(\mathbf{x}_1)}\left[D_{KL}(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))||P_{2,j}(\beta_j)\right]$$

is calculable in polynomial time. Now doing this for every coordinate, we can calculate the value of the third term,

$$\sum_{j=1}^{m_2}\mathbb{E}_{P_1(\mathbf{x}_1)}\left[D_{KL}(Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))||P_{2,j}(\beta_j)\right]$$

**Second term of** (6)**:** $R_2 = \frac{n_2}{2}\log(2\pi\sigma_2)$

This term is constant.

**Third term of** (6)**:** $R_3 = \frac{1}{2\sigma_2}\mathbb{E}_{\substack{P_1(\mathbf{x}_1)\\ Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}}\left[||\mathbf{f}(\mathbf{x}_1)-\mathbf{B}_2(\mathbf{z}_2)||_2^2\right]$

We define

$$S(\mathbf{x}_2) := \mathbb{E}_{Q_2(\mathbf{z}_2|\mathbf{x}_2)}\left[\mathbf{B}_2\mathbf{z}_2\right]$$

and

$$D(\mathbf{x}_2) := \mathbb{E}_{Q_2(\mathbf{z}_2|\mathbf{x}_2)}\left[||\mathbf{B}_2\mathbf{z}_2-\mathbb{E}[\mathbf{B}_2\mathbf{z}_2]||_2^2\right] = \mathrm{tr}(\mathrm{Cov}_{Q_2(\mathbf{z}_2|\mathbf{x}_2)}(\mathbf{B}_2\mathbf{z}_2))$$

Applying these notations, the third term of (6) is equal to

$$2\sigma_2R_3 = \mathbb{E}_{\substack{P_1(\mathbf{x}_1)\\ Q_2(\mathbf{z}_2|\mathbf{f}(\mathbf{x}_1))}}\left[||\mathbf{f}(\mathbf{x}_1)-\mathbf{B}_2(\mathbf{z}_2)||_2^2\right] = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[D(\mathbf{f}(\mathbf{x}_1))\right]+\mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{f}(\mathbf{x}_1)-S(\mathbf{f}(\mathbf{x}_1))||_2^2\right] \tag{7}$$

*First term of* (7)*:* $S_1 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[D(\mathbf{f}(\mathbf{x}_1))\right]$

Since we assumed that all $\beta_j$ are independent,

$$D(\mathbf{f}(\mathbf{x}_1)) = \mathrm{tr}(\mathrm{Cov}_{Q_2(\mathbf{z}_2|\mathbf{x}_2)}(\mathbf{B}_2\mathbf{z}_2)) = \sum_{1\le j\le m_2}\mathbf{H}_{j,j}\mathrm{Var}_{Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))}(\beta_j)$$

where $\mathbf{H} = \mathbf{B}_2^T\mathbf{B}_2$.

According to Lemma 3.4, we can determine the entire distribution of $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$, and for all $(\frac{T}{\varepsilon})^t$ values of $\mathbf{C}_j\mathbf{f}(\mathbf{x}_1)$, we can calculate the $Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1)) = Q'_{2,j}(\beta_j|\mathbf{C}_j\mathbf{f}(\mathbf{x}_1))$ distribution and its variance for all $j$ in $O(\frac{T}{\varepsilon})$ time. Thus, we can calculate in polynomial time the value of $\mathbb{E}_{P_1(\mathbf{x}_1)}\left[D(\mathbf{f}(\mathbf{x}_1))\right]$.

*Second term of* (7)*:* $S_2 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{f}(\mathbf{x}_1)-S(\mathbf{f}(\mathbf{x}_1))||_2^2\right]$

Let's call the $j$th column of the $\mathbf{B}_2$ matrix $\mathbf{u}_j$, and let

$$s_j(\mathbf{x}_2) := \mathbb{E}_{Q_{2,j}(\beta_j|\mathbf{f}(\mathbf{x}_1))}[\beta_j]$$

Then

$$S(\mathbf{x}_2) = \mathbb{E}_{Q_2(\mathbf{z}_2|\mathbf{x}_2)}\left[\mathbf{B}_2\mathbf{z}_2\right] = \mathbf{B}_2\left(\mathbb{E}_{Q_2(\mathbf{z}_2|\mathbf{x}_2)}[\mathbf{z}_2]\right) = \sum_{j=1}^{m_2}s_j(\mathbf{x}_2)\mathbf{u}_j.$$

We decompose $S_2$ as

$$S_2 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{f}(\mathbf{x}_1) - S(\mathbf{f}(\mathbf{x}_1))||_2^2\right] = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{f}(\mathbf{x}_1)||_2^2\right] - 2\sum_{j=1}^{m_2}\mathbb{E}_{P_1(\mathbf{x}_1)}[\langle \mathbf{f}(\mathbf{x}_1), \mathbf{u}_j\rangle s_j(\mathbf{f}(\mathbf{x}_1))] + \sum_{j,k=1}^{m_2}\mathbb{E}_{P_1(\mathbf{x}_1)}\left[s_j(\mathbf{f}(\mathbf{x}_1))s_k(\mathbf{f}(\mathbf{x}_1))\langle \mathbf{u}_j, \mathbf{u}_k\rangle\right] \tag{8}$$

*First term of* (8)*:* $T_1 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{f}(\mathbf{x}_1)||_2^2\right]$

$$\begin{aligned} T_1 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{f}(\mathbf{x}_1)||_2^2\right] &= \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{A}\mathbf{x}_1 + \psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)\mathbf{u}||_2^2\right] \\ &= \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{A}\mathbf{x}_1||_2^2\right] + 2\mathbb{E}_{P_1(\mathbf{x}_1)}\left[\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle \cdot \psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)\right] \\ &\quad + \mathbb{E}_{P_1(\mathbf{x}_1)}\left[\psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)^2||\mathbf{u}||_2^2\right] \end{aligned} \tag{9}$$

*First term of* (9)*:* $U_1 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{A}\mathbf{x}_1||_2^2\right]$

Notice that

$$\mathbf{A}\mathbf{x}_1 = \mathbf{A}\mathbf{B}_1\mathbf{z}_1 + \mathbf{A}\varepsilon = \sum_{j=1}^{m_1}\alpha_j\mathbf{r}_j + \mathbf{A}\boldsymbol{\zeta}$$

where $\mathbf{r}_j$ vectors are the columns of the $\mathbf{A}\mathbf{B}_1$ matrix, $\alpha_j$ are independently sampled from the respective $P_{1,j}(\alpha_j)$ distributions and $\mathbf{A}\varepsilon$ is a normal distribution with known covariance matrix that's independent of everything else. Therefore we can easily compute

$$U_1 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[||\mathbf{A}\mathbf{x}_1||_2^2\right] = ||\mathbb{E}_{P_1(\mathbf{x}_1)}\left[\mathbf{A}\mathbf{x}_1\right]||_2^2 + \text{tr}(\text{Cov}_{P_1(\mathbf{x}_1)}(\mathbf{A}\mathbf{x}_1))$$

using

$$\mathbb{E}_{P_1(\mathbf{x}_1)}\left[\mathbf{A}\mathbf{x}_1\right] = \sum_{j=1}^{m_1}\mathbf{r}_j\mathbb{E}_{P_{1,j}}[\alpha_j]$$

and

$$\text{tr}(\text{Cov}_{P_1(\mathbf{x}_1)}(\mathbf{A}\mathbf{x}_1)) = \text{tr}(\text{Cov}(\mathbf{A}\boldsymbol{\zeta})) + \sum_{j=1}^{m_1}\text{Var}_{P_{1,j}}(\alpha_j)||\mathbf{r}_j||_2^2.$$

*Second term of* (9)*:* $U_2 = 2\mathbb{E}_{P_1(\mathbf{x}_1)}\left[\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle \cdot \psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)\right]$

We already established in the proof of Lemma 3.4 that we can calculate the joint distribution of the dot product of $\mathbf{A}\mathbf{x}_1$ with arbitrary vectors. So we can calculate the joint distribution of $\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle$ and $\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle$, from which we can calculate

$$U_2 = 2\mathbb{E}_{P_1(\mathbf{x}_1)}\left[\langle \mathbf{A}\mathbf{x}_1, \mathbf{u}\rangle \cdot \psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)\right]$$

*Second term of* (9)*:* $U_3 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[\psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)^2||\mathbf{u}||_2^2\right]$.

As above, we can calculate the distribution of $\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle$, so we can determine the value of

$$U_3 = \mathbb{E}_{P_1(\mathbf{x}_1)}\left[\psi(\langle \mathbf{A}\mathbf{x}_1, \mathbf{w}\rangle)^2||\mathbf{u}||_2^2\right].$$

This concludes the calculation of (9), and now we look at the other terms in (8):

*Second term of* (8)*:* $T_2 = -2 \sum_{j=1}^{m_2} \mathbb{E}_{P_1(\mathbf{x}_1)}[\langle \mathbf{f}(\mathbf{x}_1), \mathbf{u}_j \rangle s_j(\mathbf{f}(\mathbf{x}_1))]$

Given $\mathbf{C}_j \mathbf{f}(\mathbf{x}_1)$, we can calculate the $Q_{2,j}(\beta_j | \mathbf{f}(\mathbf{x}_1))$ distribution, so we can calculate its expectation $s(\mathbf{f}(\mathbf{x}_1))$ too. By Lemma 3.4, we can determine the joint distribution of the $t$ coordinates of $\mathbf{C}_j \mathbf{f}(\mathbf{x}_1)$ and $\langle \mathbf{f}(\mathbf{x}_1), \mathbf{u}_j \rangle$. That's all we need to calculate an $\mathbb{E}_{P_1(\mathbf{x}_1)}[\langle \mathbf{f}(\mathbf{x}_1), \mathbf{u}_j \rangle s_j(\mathbf{f}(\mathbf{x}_1))]$ term. There are $m_2$ terms like this, so calculating and summing them all still takes polynomial time.

*Third term of* (8)*:* $T_3 = \sum_{j,k=1}^{m_2} \mathbb{E}_{P_1(\mathbf{x}_1)} \left[ s_j(\mathbf{f}(\mathbf{x}_1)) s_k(\mathbf{f}(\mathbf{x}_1)) \langle \mathbf{u}_j, \mathbf{u}_k \rangle \right]$

We can also calculate the joint distribution of $\mathbf{C}_j \mathbf{f}(\mathbf{x}_1)$ and $\mathbf{C}_j \mathbf{f}(\mathbf{x}_1)$ for any $j$ and $k$, and that's what we need to calculate the $\mathbb{E}_{P_1(\mathbf{x}_1)} \left[ s_j(\mathbf{f}(\mathbf{x}_1)) s_k(\mathbf{f}(\mathbf{x}_1)) \langle \mathbf{u}_j, \mathbf{u}_k \rangle \right]$ terms. Calculating and summing $m_2^2$ of them still takes polynomial time. This is the final missing term of $S_2$, so we can calculate it in polynomial time. This concludes the computation of $R_3$, therefore we have everything to compute the desired $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ too.

□

## 4 Conclusion

We have given a novel framework for learning representations for the activations of a particular form of neural network. The learning procedure for this representation is analytically tractable, in the sense that layer-by-layer we computed the loss associated to the new parameters in terms of previously learned parameters and the coefficients of the transition functions (but crucially not requiring any true activations of the model via sampling). The distribution on each layer is modelled as a sum of independent features with Gaussian noise which limits the expressive power of our representation, but we hope that in future more flexible representations will be found that still bypass the need for sampling.

## 5 Discussion

This exercise was an existence proof of the applicability of analytical calculations for settings that are more complicated than our previous work [1] on Gaussians. However, using the above presented approach for modelling the activations of a neural net has several drawbacks. We list some of the main obstacles to this approach working well in practice:

1. The above presented method heavily relies on the starting distribution of $P_1(\mathbf{x}_1)$ being the sum of independent features with some additional Gaussian noise. Although the Linear Representation Hypothesis is a common assumption in much of the existing interpretability work, our assumption of independence among the linear features is a much stronger assumption.

   Since there are many distributions that cannot be well approximated as the sum of independent features, this significantly decreases the reliability of our activation modelling until we can find richer distributions for which a similar analytic method can apply.
2. This method as presented here can only deal with transition functions $\mathbf{f}$ that apply non-linearity in only one direction. Alhough we can build MLPs from transition functions like this, building a single real layer requires as many layers in our model as the width of the network. Trying to follow the modelling through this many layers will in practice lead to an unacceptable amount of accumulated error.
3. Minimizing the VAE loss defined in Lemma 2.2 leads to imperfect models even in the case of very simple $f$ functions. Notably, if $f$ is the identity, and we have an exact description of $P_1(\mathbf{x}_1)$, it would be reasonable to expect it from our activation model to perfectly reconstruct the the distribution of $P_2(\mathbf{x}_2)$ as being the same as the distribution of $P_1(\mathbf{x}_1)$. However, one can show that this is not exactly true in our case, even if we can use an arbitrarily broad family to select the $Q_{2,j}$ distributions from. The best example to check is the one where $m_1 = n_1 = m_2 = n_2 = 2$, and $P_{1,1}$ and $P_{1,2}$ are both Bernoulli-distributions and $\mathbf{B}_1$'s columns (the feature vectors) subtend a small angle. Then one can empirically check that our method doesn't perfectly reconstruct the original distribution.

4. As we prove in Lemma 2.2,

$$H(P_1(\mathbf{f}^{-1}(\mathbf{x}_2)), P_{2,\boldsymbol{\theta}}(\mathbf{x}_2)) = -\mathbb{E}_{\mathbf{x}_1 \sim P_1}\left[\log\left(P_{2,\boldsymbol{\theta}}\left(\mathbf{f}(\mathbf{x}_1)\right)\right] \leq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})\right.$$

   The whole reason we chose this loss function is that minimizing it gives us an upper bound on the cross-entropy of our model of the second layer, $P_{2,\boldsymbol{\theta}}(\mathbf{x}_2)$, with the true distribution, $P_1(f^{-1}(\mathbf{x}_2))$.
   However, we have no guarantees [6] bounding $H(P_1, P_3)$ given $H(P_1, P_2)$ and $H(P_2, P_3)$. Thus, when we use our method repeatedly to model the next layer of a neural net based on the model of the previous layer, we only have extremely weak guaranteed upper bound for how far we eventually stray from the true distribution of outputs.

5. The original reason we were interested in using analytical methods for activation modelling was that we wanted to estimate low probability events. While we showed an example of analytic estimations working, the presented method of bounding the cross-entropy of distributions doesn't particularly attend to low probability events. Two distributions can have very low cross-entropy, while the probability they assign to a tail event can be drastically different. To strengthen the method, we will need to look for ways that don't just try to generically model the distribution, but put special focus on modelling the parts that are relevant to the question we want to decide. We currently consider this one of the most important open questions in our work.

## References

[1] Paul Christiano, Eric Neyman, and Mark Xu. Formalizing the presumption of independence, 2022.

[2] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

[3] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models, 2019.

[4] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation, 2015.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[6] Solomon Kullback. Information theory and statistics p. 6, 1959.